

Computers: Software

Patrice Koehl
Computer Science
UC Davis

Acknowledgments

Thanks to the following web site for the images used in this presentation:

- Wikipedia
- <http://microsoft.toddverbeek.com>
- <http://www.webopedia.com>
- <http://www.engin.umd.umich.edu/>
- <http://www.dell.com>
- <http://www.intel.com>
- <http://www.apple.com>
- <http://www.ibm.com>
- http://homepages.feis.herts.ac.uk/~msc_ice/unit2/
- <http://www.howstuffworks.com>

Computer Layers

Hardware

BIOS

Operating System

Software

Programming languages



BIOS: Basic Input/Output Layer

BIOS refers to the **firmware code** usually stored on the **PROM, EPROM** or flash drive that is run by a computer when first powered on.

BIOS performs three major tasks:

- First, the **Power On Self Tests (POST)** are conducted. These tests verify that the hardware system is operating correctly.
- Second, the BIOS **initiates** the different hardware component of the system, scanning their own ROM or PROM.
- Third, the BIOS initiate the **boot process**. The BIOS looks for boot information that is contained in file called the **master boot record (MBR)** at the first sector on the disk. Once an acceptable boot record is found the **operating system** is loaded which takes over control of the computer.

Computer Layers

Hardware

BIOS

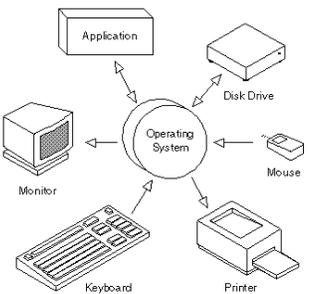
Operating System

Software

Programming languages



The operating system



```
graph TD; OS((Operating System)) --- Application[Application]; OS --- DiskDrive[Disk Drive]; OS --- Mouse[Mouse]; OS --- Printer[Printer]; OS --- Keyboard[Keyboard]; OS --- Monitor[Monitor];
```

The operating system

Definition found on Wikipedia:

“An operating system (OS) is the software that manages the sharing of the resources of a computer and provides programmers with an interface used to access those resources”

Most common operating systems:

- DOS (desktops, laptops)
- Unix and variants, including Linux (servers)
- MacOS

The operating system

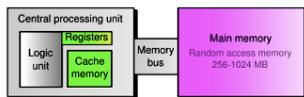
Operating systems can be classified as follows:

- multi-user : Allows two or more users to run programs at the same time.
- multiprocessing : Supports running a program on more than one CPU.
- multitasking : Allows more than one program to run concurrently.
- multithreading : Allows different parts of a single program to run concurrently.

The operating system

Memory management:

Current computers organize memory resources hierarchically, from registers, CPU cache, RAM and disks.



The virtual memory manager coordinates the use of these resources by tracking which one is available, which is to be allocated or deallocated and how to move data between them.

If running processes require significantly more RAM than is available, the system may start thrashing.

The operating system

Most **operating systems** come with an application that provides a **user interface** for managing the operating system, such as a command line interpreter or **graphical user interface (GUI)**.

Operating systems provide a **software platform** on top of which other programs, called **application programs**, can run.

Your choice of operating system determines the applications you can run.

Computer Layers

Hardware

BIOS

Operating System

Software

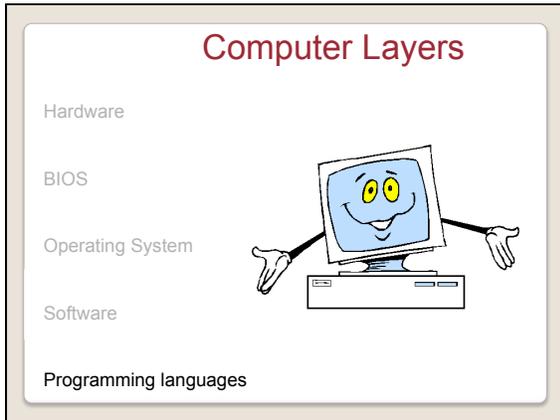
Programming languages



Application Software

Application software is a class of **computer software** that uses the capabilities of a computer to a task that the user wishes to perform. The term application refers to both the application software and its implementation.

Typical examples of software applications are **word processors**, **spreadsheets**, and **media players**.



Programming languages

A **programming language** is an **artificial language** that can be used to control the **behavior** of a machine, particularly a computer.

Programming languages are used to **facilitate communication** about the task of **organizing** and **manipulating information**, and to express **algorithms** precisely.

(An **algorithm** is a list of well-defined **instructions** for completing a task; that is, given an initial state, it will proceed through a well-defined series of successive states, eventually terminating in an end-state. The transition from one state to the next is not necessarily **deterministic**; some algorithms, known as **probabilistic algorithms**, incorporate **randomness**)

Programming languages

There are many, many programming languages, and new ones appear every year.

(<http://www.engin.umd.umich.edu/CIS/course.des/cis400/>)

Programming languages

Three main levels of programming languages:

-Machine languages: refers to the "ones and zeroes" that processors use as instructions. Give it one pattern of bits (such as 11001001) and it will add two numbers, give it a different pattern (11001010) and it will instead subtract one from the other.

-Assembly languages: is as close as you can come to writing in machine language, but has the advantage that it's also human-readable... using a small vocabulary of words with one syllable, such as:
MOV A, B

-High level languages: A vocabulary and set of grammatical rules for instructing a computer to perform specific tasks. Each language has its own set of keywords and its own syntax.

Programming languages: Interpret or Compile?

Regardless of what language you use, you eventually need to convert your program into machine language so that the computer can understand it. There are two ways to do this:

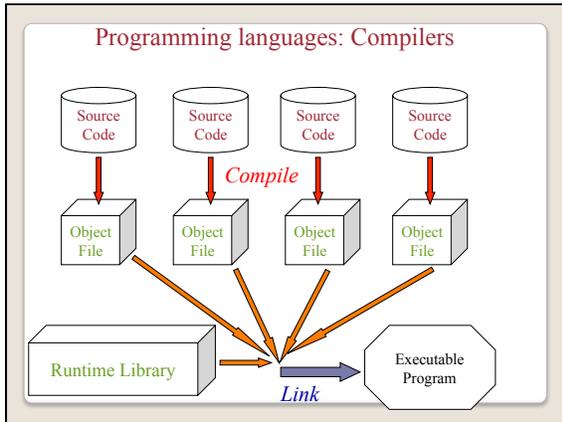
- interpret the program through an interpreter
- compile the program through a compiler

The main disadvantage of interpreters is that when a program is interpreted, it runs slower than if it had been compiled.

Programming languages: Interpreters

An interpreter is a program that either:

- executes the source code directly (type I)
- translates source code into some efficient intermediate representation and immediately executes this (type II)
- is invoked to explicitly execute stored precompiled code made by a compiler which is part of the interpreter system (type III)



Programming languages: Compilers

A **compiler** is a program that **translates source codes** into **object codes**. The compiler derives its name from the way it works, looking at the entire source code and collecting and reorganizing the instructions.

Thus, a **compiler** differs from an **interpreter**, which analyzes and executes each line of source code successively, without analyzing the entire program.

Programming languages: Examples

Interpreted languages:

- Perl, Python, Matlab (type II)
- Java (type III)

Compiled languages:

- Fortran
- C, C++
- Pascal
- Basic
- Cobol
- ADA
